CSC 261/461

Query Optimization

Introduction

- Query optimization
  - Conducted by a query optimizer in a DBMS
  - Goal: select best available strategy for executing query
    - Based on information available
- Most RDBMSs use a **tree** as the internal representation of a query

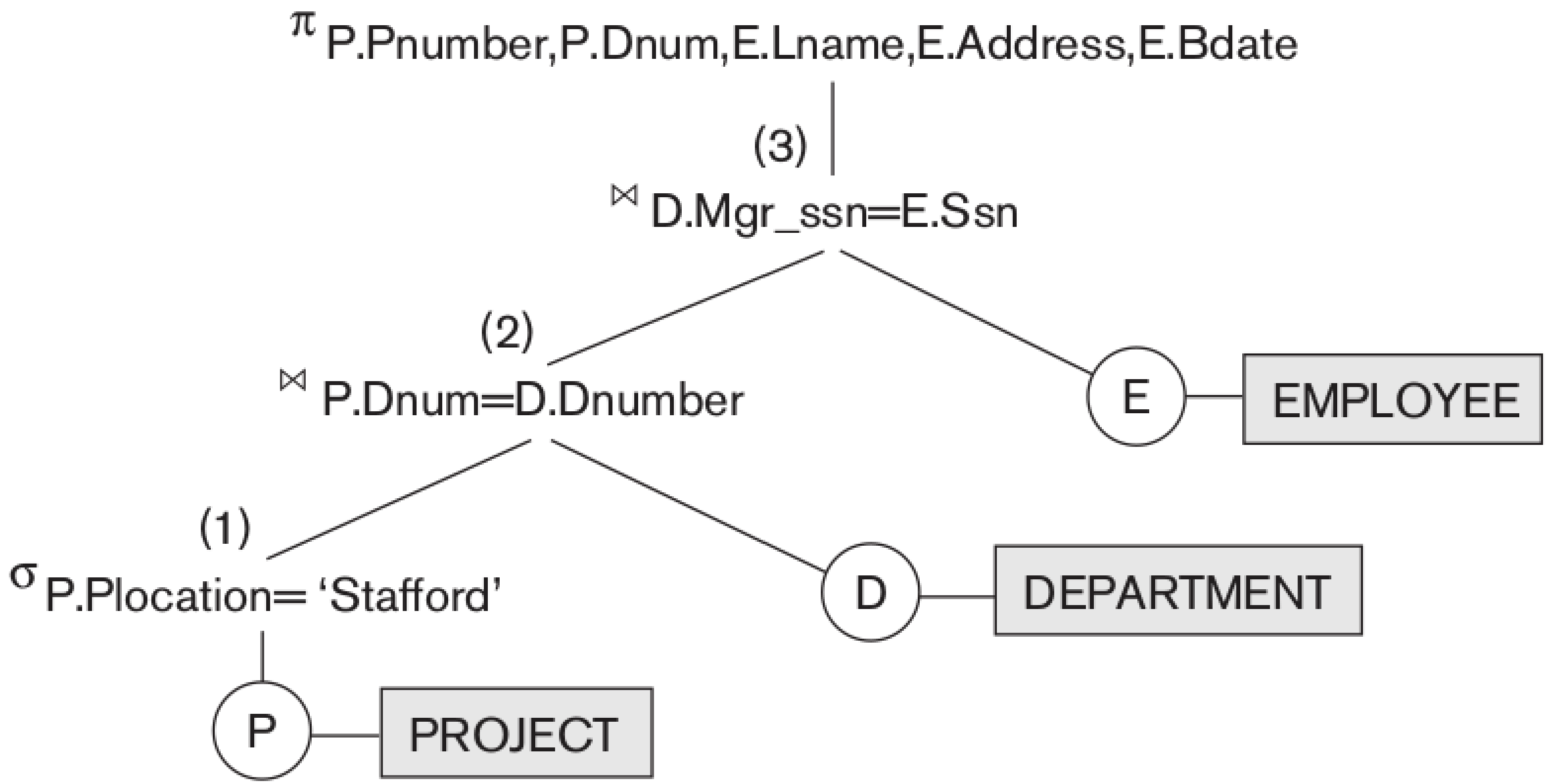
Query Trees and Heuristics for Query Optimization

- Step 1: scanner and parser generate initial query representation
- Step 2: representation is optimized according to heuristic rules
- Step 3: query execution plan is developed
  - Execute groups of operations based on access paths available and files involved

Query Trees and Heuristics for Query Optimization

- A **query tree** is a tree data structure that corresponds to a relational algebra expression.
- Input relations of the query are *leaf nodes* of the tree
- RA operations are *internal nodes*.
- An **execution** of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation.
- Execution start from leaves.

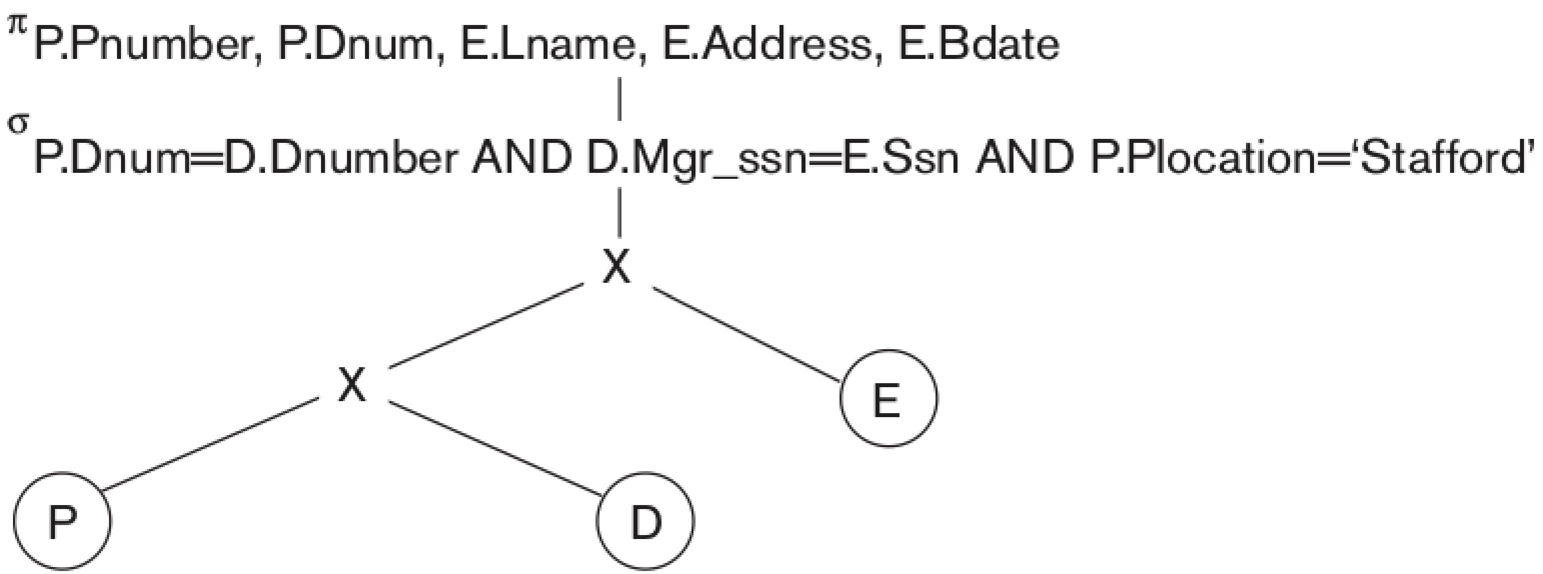
Query Trees



$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$

Heuristic Optimization of Query Trees

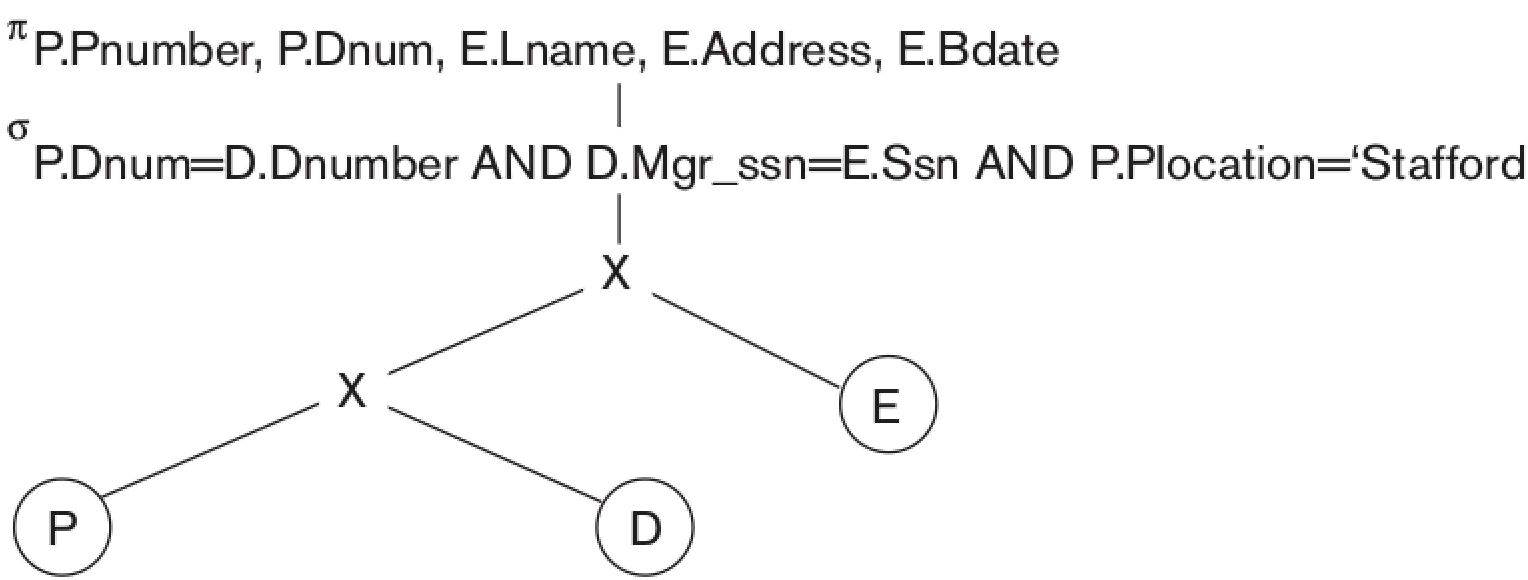
- Many different query trees can be used to represent the query and get the same results
- Initial tree for Q2
  - Very inefficient - will never be executed
  - Optimizer will transform into equivalent final query tree



Heuristic Optimization of Query Trees

- Query

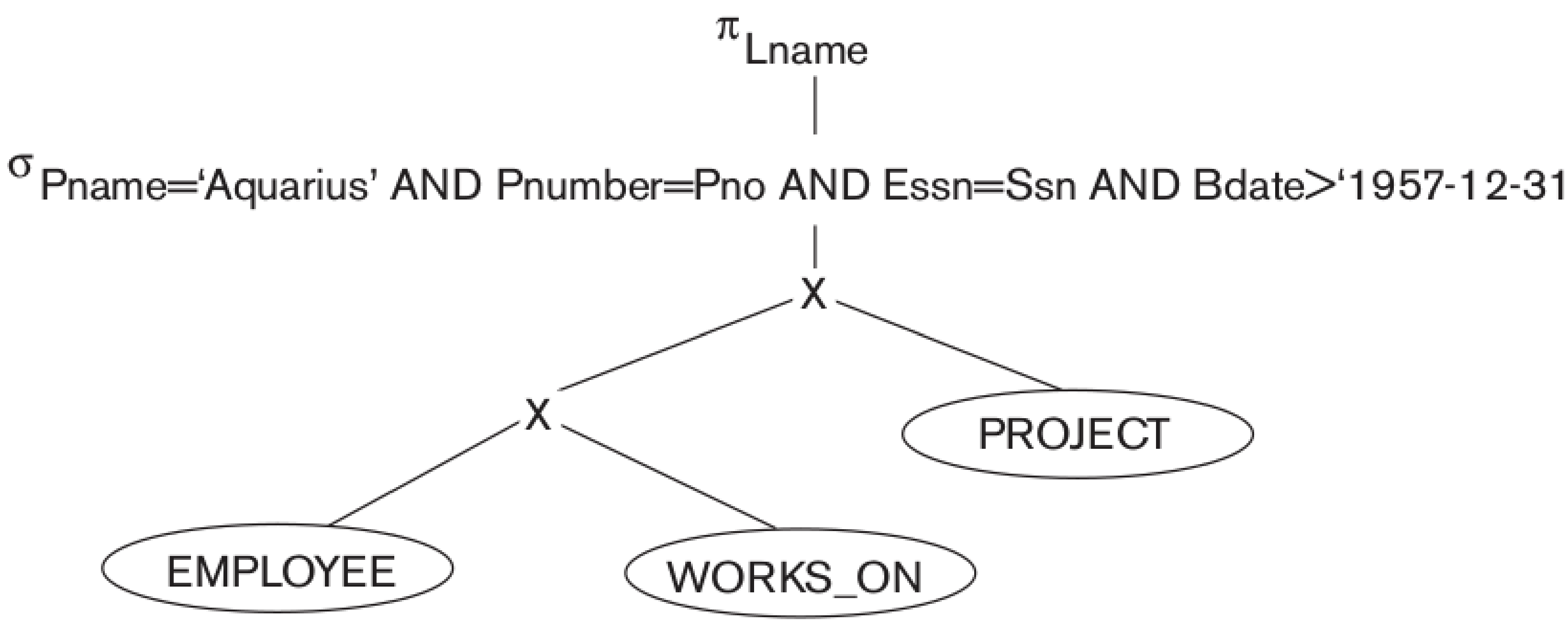
```
Q2:  SELECT  P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate
     FROM    PROJECT P, DEPARTMENT D, EMPLOYEE E
     WHERE   P.Dnum=D.Dnumber AND D.Mgr_ssn=E.Ssn AND
             P.Plocation= 'Stafford';
```



Query Transformation Example

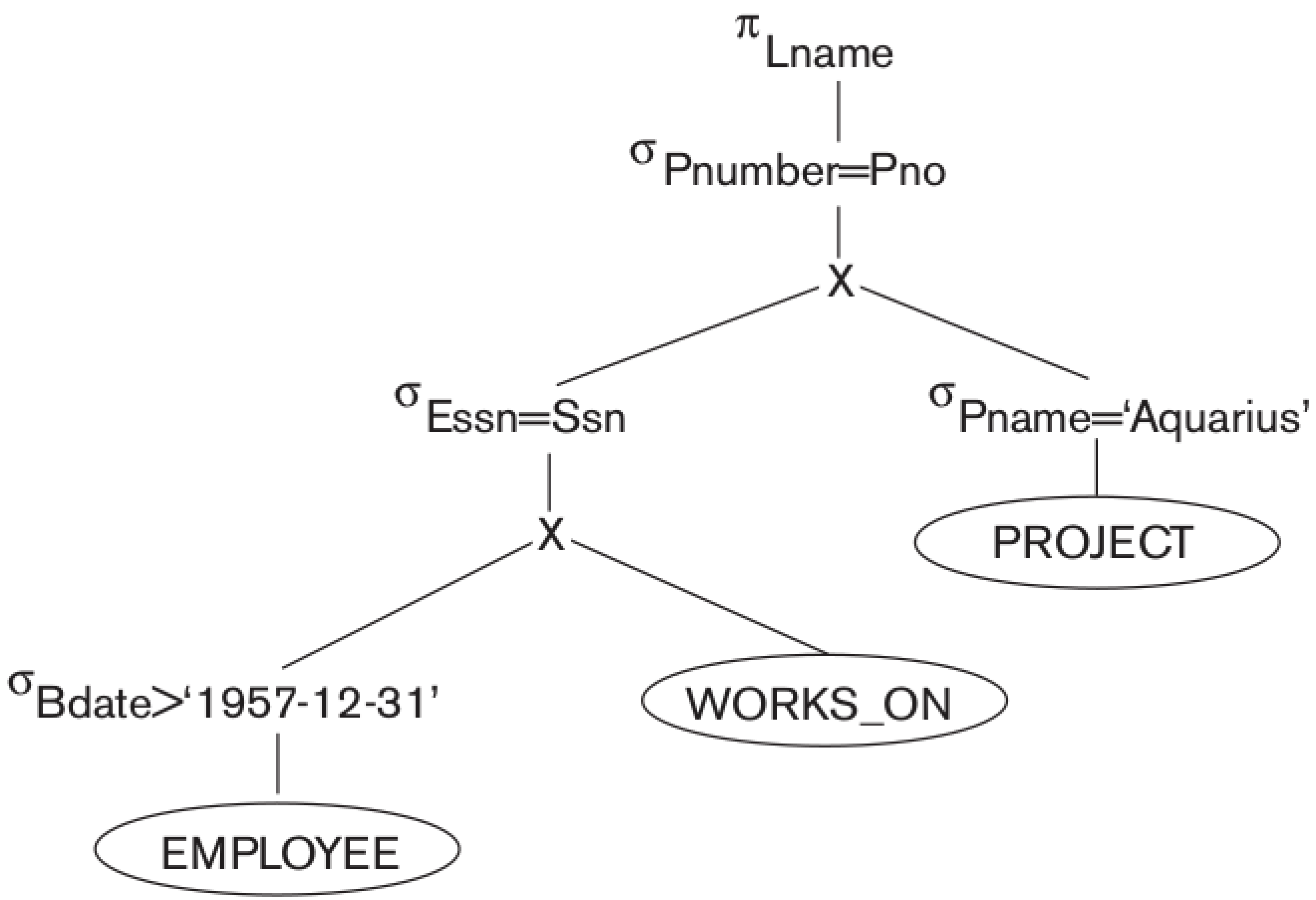
- Q: *Find the last names of employees born after 1957 who work on a project named 'Aquarius'*

```
SELECT  E.Lname
FROM    EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE   P.Pname='Aquarius' AND P.Pnumber=W.Pno AND E.Essn=W.Ssn
        AND E.Bdate > '1957-12-31';
```



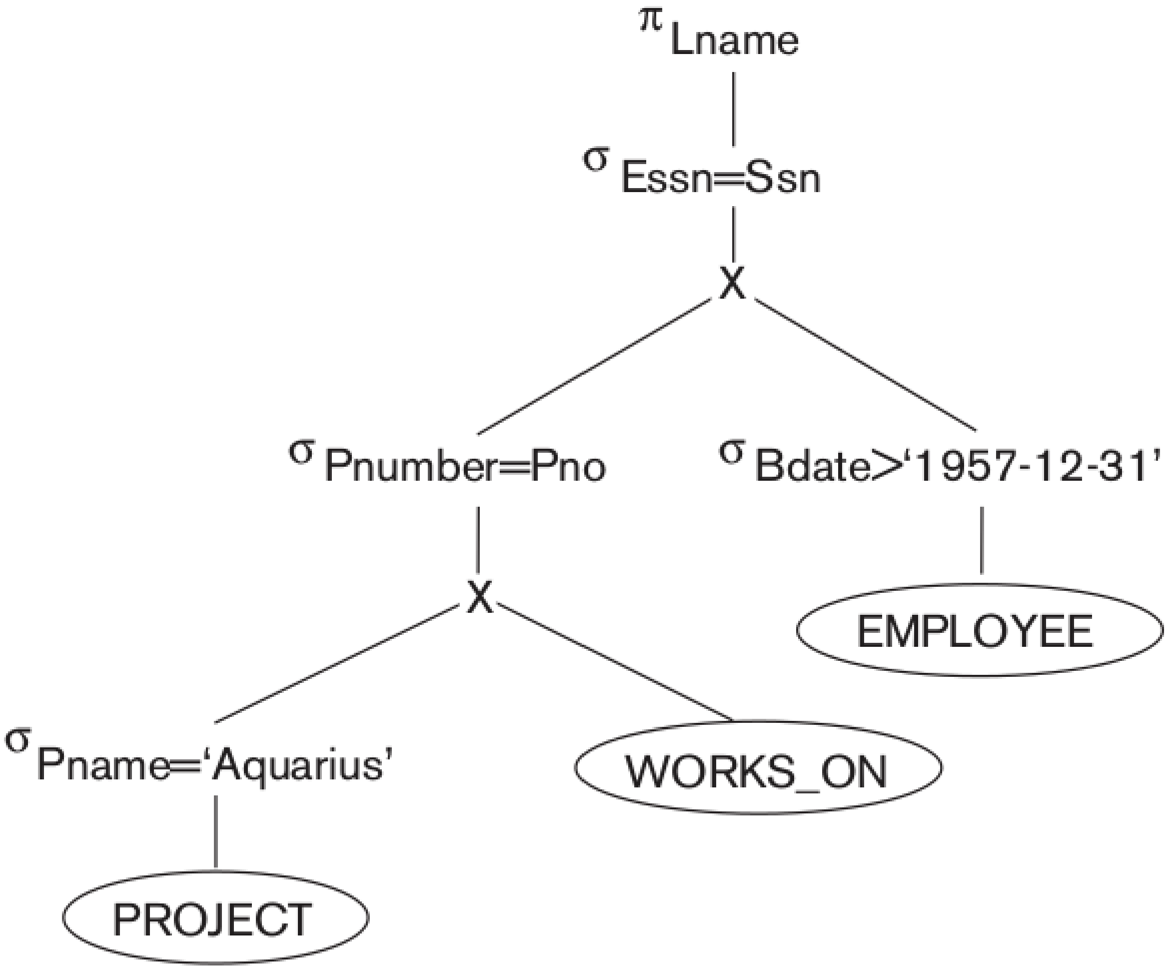
Query Transformation Example (cont'd.)

First improvement



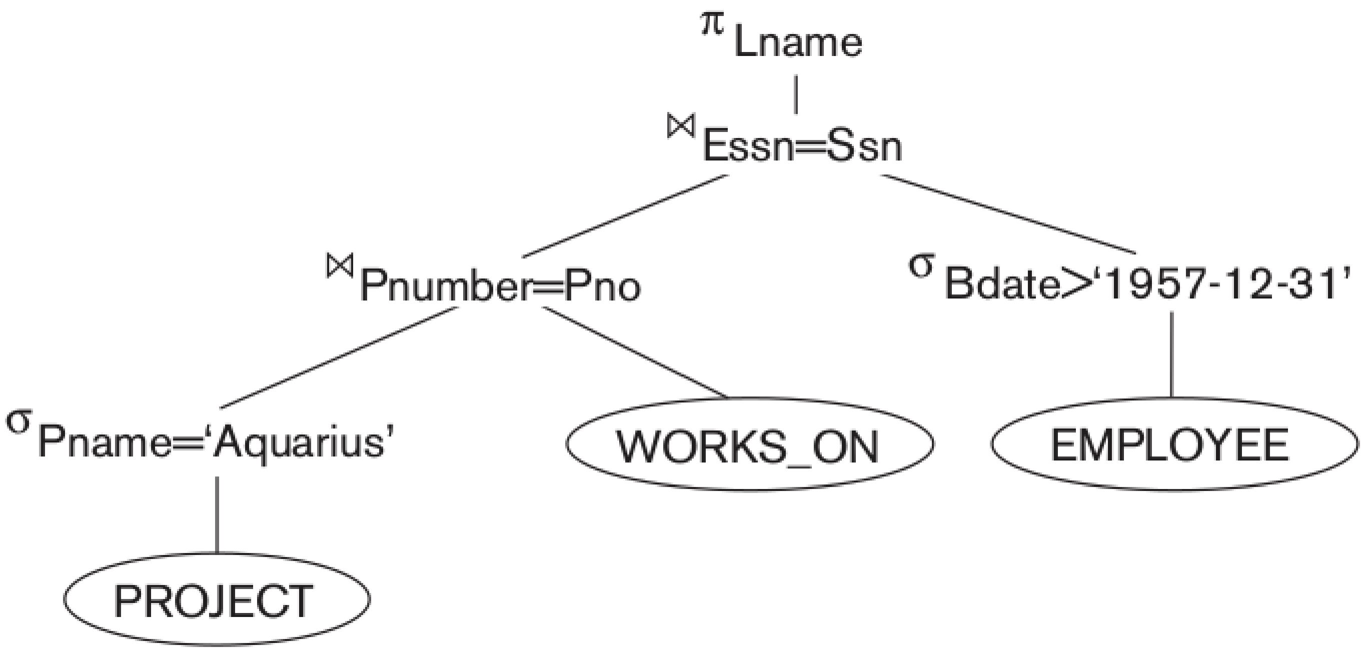
Query Transformation Example (cont'd.)

Second improvement



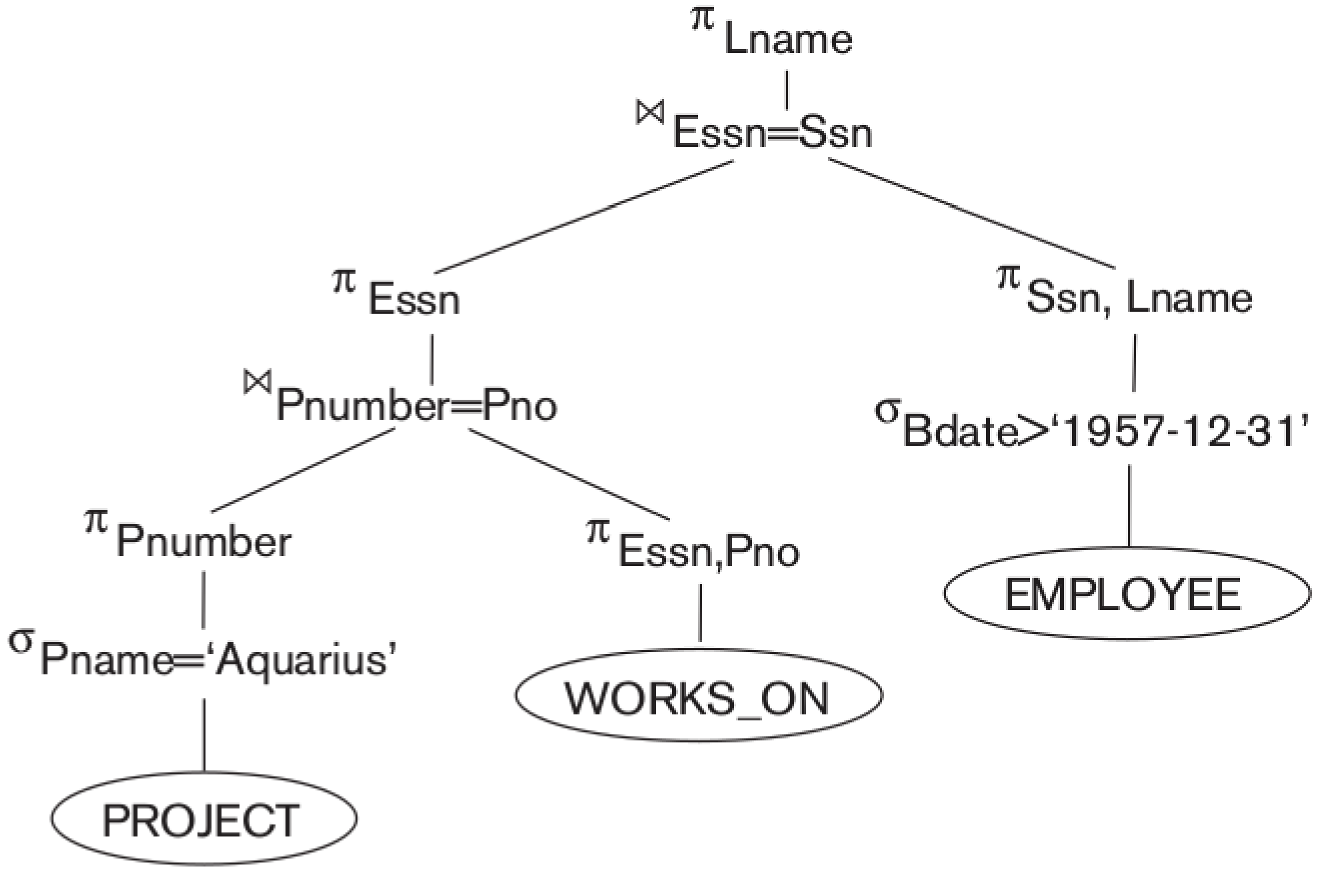
Query Transformation Example (cont'd.)

Third improvement



Query Transformation Example (cont'd.)

Next improvement



General Transformation Rules for RA Equations

Transformation rules are useful in query optimization

- **R1 Cascade of σ**. A conjunctive selection condition can be broken up into a cascade of individual σ operations

$$\sigma_{c_1 \text{ AND } c_2 \text{ AND } \ldots \text{ AND } c_n}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(\ldots(\sigma_{c_n}(R))\ldots))$$

- **R2 Commutativity of σ**. The σ operation is commutative:

$$\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$$

- **R3 Cascade of π**. In a cascade (sequence) of π operations, all but the last one can be ignored:

$$\pi_{\text{List}_1}(\pi_{\text{List}_2}(\ldots(\pi_{\text{List}_n}(R))\ldots)) \equiv \pi_{\text{List}_1}(R)$$

General Transformation Rules for RA Equations

- **R4 Commuting σ with π.** If the selection condition *c* involves only attributes in the projection list, operations can be commuted:

$$\pi_{A_1, A_2, \ldots, A_n} (\sigma_c (R)) \equiv \sigma_c (\pi_{A_1, A_2, \ldots, A_n} (R))$$

- **R5 Commutativity of *join*.** The join operation is commutative, as is the × operation:

$$R \bowtie_c S \equiv S \bowtie_c R$$
$$R \times S \equiv S \times R$$

General Transformation Rules for RA Equations

- **R6 Commuting σ with *join*.**
  - If attributes in *c* involve only attributes of one of the relations being joined (R) the two operations can be commuted as follows:

$$\sigma_c\,(R \bowtie S) \equiv (\sigma_c\,(R)) \bowtie S$$

  - if *c* is (c1 AND c2), where c1 involves only attributes of R and c2 involves only attributes of S:

$$\sigma_c\,(R \bowtie S) \equiv (\sigma_{c_1}\,(R)) \bowtie (\sigma_{c_2}\,(S))$$

General Transformation Rules for RA Equations

- **R7 Commuting π with *join*.** If L = {A1 , ... , An, B1 , ... , Bm}, with Ai in R and Bi in S.
  - If *c* involves only attributes in L:

$$\pi_L (R \bowtie_c S) \equiv (\pi_{A_1, \ldots, A_n} (R)) \bowtie_c (\pi_{B_1, \ldots, B_m} (S))$$

  - If additional attributes (not in L):

$$\pi_L (R \bowtie_c S) \equiv \pi_L ((\pi_{A_1, \ldots, A_n, A_{n+1}, \ldots, A_{n+k}}(R)) \bowtie_c (\pi_{B_1, \ldots, B_m, B_{m+1}, \ldots, B_{m+p}} (S)))$$

General Transformation Rules for RA Equations

- **R8 Commutativity of set operations.** The set operations ∪ and ∩ are commutative, but − is not.
- **R9 Associativity of *join*, ×, ∪, and ∩.** These four operations are individually associative:

$$(R \ \theta \ S) \ \theta \ T \equiv R \ \theta \ (S \ \theta \ T)$$

- **R10 Commuting σ with set operations.** The σ operation commutes with ∪, ∩, and −.

$$\sigma_c \ (R \ \theta \ S) \equiv (\sigma_c \ (R)) \ \theta \ (\sigma_c \ (S))$$

- **R11 The π operation commutes with ∪.**

$$\pi_L \ (R \cup S) \equiv (\pi_L \ (R)) \cup (\pi_L \ (S))$$

General Transformation Rules for RA Equations

- **R12 Converting a (σ, ×) sequence into *join*.** If c of a σ that follows a × corresponds to a join condition:

$$(\sigma_c (R \times S)) \equiv (R \bowtie_c S)$$

- **R13 Pushing σ in conjunction with set difference.**

$$\sigma_c (R - S) = \sigma_c (R) - \sigma_c (S)$$

- **R14 Pushing σ to only one argument in ∩.** If in the condition *c* all attributes are from R, then:

$$\sigma_c (R \cap S) = \sigma_c (R) \cap S.$$

General Transformation Rules for RA Equations

- **R15 Some trivial transformations.**
  - If S is empty, then $R \cup S = R$
  - If the condition c in σ_c is true for the entire R, then $\sigma_c(R) = R$.
- **Other.** A selection or join condition *c* can be converted into an equivalent one by using rules from Boolean algebra (*De Morgan's laws*):
  - NOT (c1 AND c2) ≡ (NOT c1) OR (NOT c2)
  - NOT (c1 OR c2) ≡ (NOT c1) AND (NOT c2)

Outline of an Optimization Algorithm

- Using R1, *break up* any SELECT operations with conjunctive conditions into a cascade of SELECT operations.
  - Greater degree of freedom in moving SELECT operations down different branches of the tree.
- Using R2, 4, 6, 10, 13, 14, *move* each SELECT operation as far *down* the query tree as possible.
- Using R5 and 9, rearrange the leaf nodes of the tree using the following criteria.
  - First, position the leaf node relations with the most restrictive SELECT operations so they are executed first in the query tree representation.
  - Second, make sure that the ordering of leaf nodes does not cause CARTESIAN PRODUCT operations

Outline of an Optimization Algorithm

- Using R12, combine a CARTESIAN PRODUCT operation with a subsequent SELECT operation in the tree into a JOIN operation, if the condition represents a join condition.
- Using R3, 4, 7, and 11, break down and *move* lists of projection attributes down the tree as far as possible by creating new PROJECT operations as needed. Only those attributes needed in the query result and in subsequent operations in the query tree should be kept after each PROJECT operation.

?